MAST: Multi-Script Annotation Toolkit for Scenic Text

T Kasar, D Kumar, M N Anil Prasad, D Girish and A G Ramakrishnan
Medical Intelligence and Language Engineering Laboratory
Indian Institute of Science
Bangalore, INDIA - 560 012
{tkasar, deepak, anilprasadmn, dasarigirish, ramkiag}@ee.iisc.ernet.in

ABSTRACT

This paper describes a semi-automatic tool for annotation of multi-script text from natural scene images. The procedure involves manual seed selection followed by a region growing process to segment each word present in the image. The threshold for region growing can be varied by the user so as to ensure pixel-accurate character segmentation. The text present in the image is tagged word-by-word. A virtual keyboard interface has also been designed for entering the ground truth in ten Indic scripts, besides English. The keyboard interface can easily be generated for any script, thereby expanding the scope of the toolkit. Optionally, each segmented word can further be labeled into its constituent characters/symbols. Polygonal masks are used to split or merge the segmented words into valid characters/symbols. The ground truth is represented by a pixel-level segmented image and a '.txt' file that contains information about the number of words in the image, word bounding boxes, script and ground truth Unicode. The toolkit can be used to generate ground truth and annotation for any generic document image and hence is useful for researchers in the document image processing community for evaluating the performance of document analysis and recognition techniques.

Keywords

Annotation tool, ground truth, scene text, multi-script documents, camera-based document analysis

1. INTRODUCTION AND MOTIVATION

There is a significant need for methods to extract and recognize text in scenes. Unlike the case of processing conventional document images, natural scene text understanding usually involves a pre-processing step of text region location and extraction before subjecting the acquired image for character recognition task. Recognition is performed only on the detected text regions so as to mitigate the effects of background complexity. The availability of annotated datasets for scenic images will aid in testing and quantifying the performances of various document analysis and recognition algorithms.

The Robust Reading Competition [1] was held at the 7^{th} international conference on document analysis and recognition 2003 to find the best system able to read complete words in camera-captured images. An XML file gives the ground truth in terms of the word and character bounding boxes and their transcription. The dataset contains various kinds of degradations such as uneven lighting conditions, complex backgrounds, variable font styles, color, low resolution and text appearing on curved or shiny surfaces.

Precision and recall measures were used in the evaluation of text detection algorithms. For text localization, it is unrealistic to expect a system to agree exactly with the ground truth identified by a human tagger. So, the ICDAR 2003 robust reading competition organizers propose a rectangle-based matching to compute the precision and recall measures. However as pointed out by Wolf and Jolion [2], rectangle-based evaluation is a non-trivial task since as the detection result rarely matches perfectly with the text boundary as specified in the ground truth. It is difficult to decide on the correctness of the text detection based on bounding box measures. Moreover, character bounding boxes can have significant overlap with adjacent components especially for an inclined or curved text. This can reduce distinctive features of that character and hence affect the training process required for recognition task. Pixel-based evaluation measures are easy to calculate and easy to interpret.

A separate competition [3] has been organized recently for born-digital (web and e-mail) images as a part of ICDAR 2011 Robust Reading Competition. The ground truth is provided at the pixel level for segmentation task while bounding boxes are employed for text locating and reading tasks. However, in the ICDAR competition dataset, the orientation of the text is always horizontal. There are no samples of vertical or multi-oriented text while such text instances appear often in images. It is also limited to English. The presence of multiple scripts also require a special treatment. In a multi-lingual country like India, it is common to find English words interspersed within sentences in Indic-script documents. Many documents, forms and signboards are generally bi-lingual or multi-lingual in nature. Every script has certain distinctive characteristics and may require script-specific processing methods. Identification of script plays a vital role for the development of next generation OCRs. Besides enabling an automated processing and utilization of documents, it will also allow the use of script-specific rules to enhance the OCR output.

Recently, two datasets containing multi-script text have been created. The Char74k dataset [4] consists of English and Kannada characters segmented from scene images, handwritten text and synthetic documents. Individual characters were manually segmented and represented by its rectangular bounding boxes or polygonal segments. This dataset does not contain pixel-accurate ground truth for individual characters.

The KAIST scene text dataset [5] comprises 3000 images captured both outdoor and indoors using a digital camera or a mobile phone under different lighting conditions. The KAIST scene text database is divided into 3 categories namely Korean, English and a composite one that contains Korean as well as English. The ground truth is stored in an XML file for each image that contains information about the location of single characters or single words (using bounding boxes) and their transcription, along with global information about the image. In addition to the XML file, a bitmap image is provided where the segmentation of the text at pixel level.

Shafait et al. [6] introduce a pixel-accurate representation of documents and propose metrics to quantify over-segmentation, under-segmentation, over-segmented components, under-segmented components, missed components and false alarms for evaluating a page segmentation algorithm. Baird et al. [7] describe ground truthing policies to zone areas in document images such as machine print, handwriting, photography, blank and line arts. Suand et al. [8] presented a user-interface (UI) design for labeling elements in document images at a pixel level. The UI design is targeted specifically toward selection of collections of foreground pixels in a document image such as machine print text, machine print graphics, handwritten text, handwritten graphics, stamps and noise. After a user has selected a set of pixels with the help of the mouse, those pixels are assigned a particular color to indicate the selected label class. The label descriptions and their colors are set by a user-editable XML configuration file. The above strategies cannot be used for scene text annotation.

We seek to aid the creation of pixel-level annotated databases for research in camera-based document analysis. The software toolkit, which we call MAST, is developed for annotating multi-script textual information in scenic images. The main contributions of this paper are listed below:

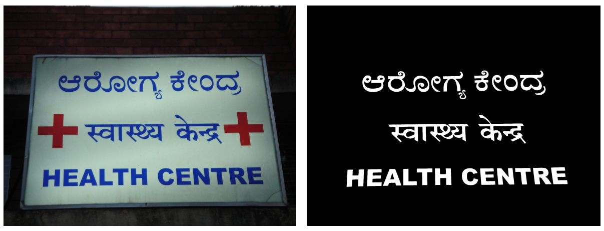
- Semi-automatic seeded region growing method that can cater to scene image. By selecting the seed points appropriately, text with color gradient and multi-colored text can also be segmented.

- The toolkit can be used to annotate multi-script documents. The virtual keyboard interface proposed in this paper can easily be extended for tagging text in any script.

- The ground truth information generated consists of a pixel-accurate segmented image, word images, script information and the unicode text that can be used for evaluating the performance of document analysis and recognition techniques.

- There is also a provision to label each character/symbol at the pixel-level that may be used in training classifiers.

- The software is open source; users can modify it to suit their needs.



(a) (b)

```
6
DSC02617_1.jpg 459 386 195 724 KANNADA ಆರೋಗ್ಯ
DSC02617_2.jpg 453 1167 169 497 KANNADA ಕೇಂದ್ರ
DSC02617_3.jpg 826 567 127 560 DEVANAGARI स्वास्थ्य
DSC02617_4.jpg 762 1195 200 334 DEVANAGARI केन्द्र
DSC02617_5.jpg 1136 276 127 731 ENGLISH HEALTH
DSC02617_6.jpg 1127 1080 125 739 ENGLISH CENTRE
```

(c)

Figure 1: (a) An example multi-script image (b) Pixel-accurate segmented image (c) The corresponding '.txt' file describing the attributes of the annotated words.

2. METHODOLOGY

Region growing technique is an ideal choice for the semi-supervised process of segmenting characters from an input image. It ensures an accurate identification of CCs due to the inherent connectivity property being invoked while grouping adjacent pixels. The input image is processed on a word-by-word basis wherein seeds are manually placed in each character of the word to initiate a region growing process. The color distance gives a measure of similarity between adjacent pixels and is used as the threshold in grouping neighboring pixels into distinct homogeneous regions. The common *RGB* color format is not suitable for color grouping tasks because it is not expressed in the way perceived by humans. Two different pairs of points with the same Euclidean distance in the *RGB* color space do not result in the same change in the perceived color. So, we use a uniform color space, namely CIE $L^*a^*b^*$, in which similar changes in color distance correspond to similar recognizable changes in the perceived color. The distance between the colors $\mathrm{C}_1 = (L_1^*, a_1^*, b_1^*)^T$ and $\mathrm{C}_2 = (L_2^*, a_2^*, b_2^*)^T$ in the $L^*a^*b^*$ color space is given by the Euclidean distance:

$$\mathrm{Dist}(\mathrm{C}_1, \mathrm{C}_2) = \sqrt{(L_1^* - L_2^*)^2 + (a_1^* - a_2^*)^2 + (b_1^* - b_2^*)^2} \quad (1)$$

The threshold parameter can be interactively varied by the user depending on the image till the word is accurately segmented. For storing the ground truth, a directory with the same filename as that of the input image is created. The segmented words, pixel-level segmented image and a '.txt' file containing the number of words, filename for saving the segmented word, corresponding bounding box, script label and the unicode text are stored in the image directory itself.

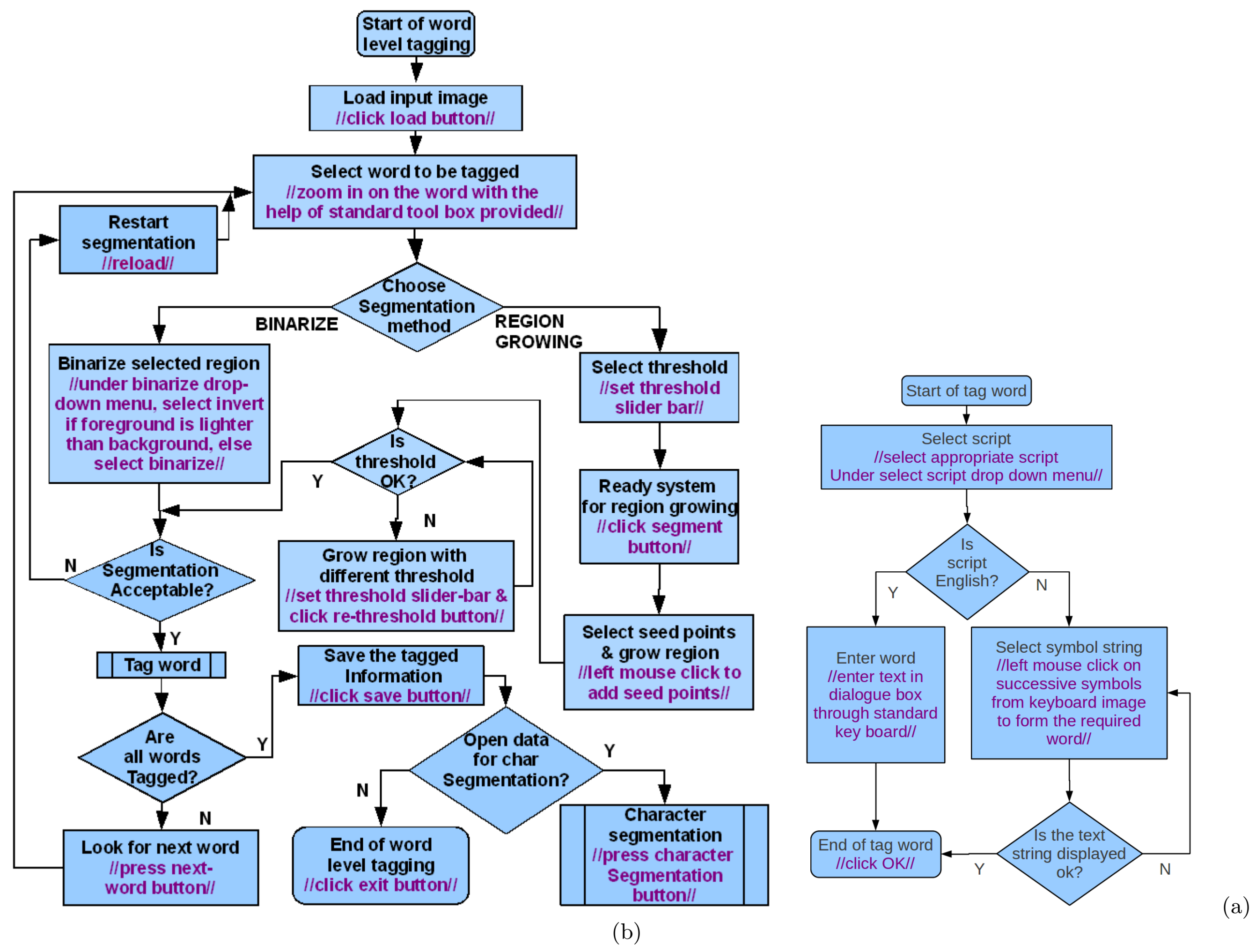

Start of word level tagging
Load input image //click load button//
Select word to be tagged //zoom in on the word with the help of standard tool box provided//
Restart segmentation //reload//
Choose Segmentation method
BINARIZE
REGION GROWING
Binarize selected region //under binarize drop-down menu, select invert if foreground is lighter than background, else select binarize//
Select threshold //set threshold slider bar//
Is threshold OK?
Y
N
Ready system for region growing //click segment button//
Grow region with different threshold //set threshold slider-bar & click re-threshold button//
Is Segmentation Acceptable?
N
Y
Select seed points & grow region //left mouse click to add seed points//
Tag word
Save the tagged Information //click save button//
Are all words Tagged?
Y
N
Open data for char Segmentation?
N
Y
Look for next word //press next-word button//
End of word level tagging //click exit button//
Character segmentation //press character Segmentation button//
(b)
Start of tag word
Select script //select appropriate script Under select script drop down menu//
Is script English?
Y
N
Enter word //enter text in dialogue box through standard key board//
Select symbol string //left mouse click on successive symbols from keyboard image to form the required word//
End of tag word //click OK//
Is the text string displayed ok?
Y
N
(a)

Figure 2: (a) Schematic block diagram of MAST for word-level annotation. (b) Flowchart for tagging a segmented word.

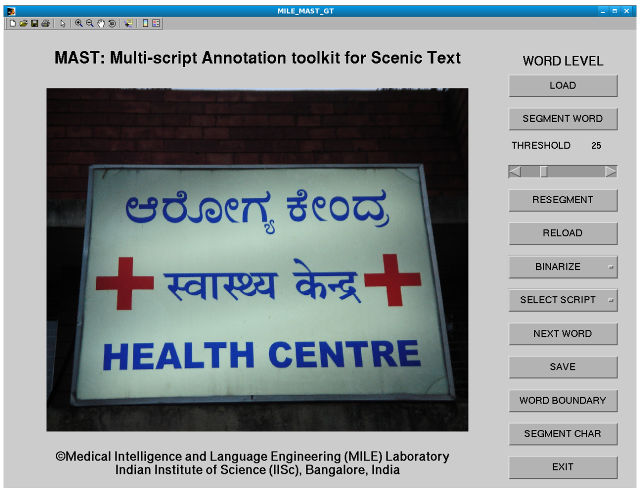


Figure 3: Screenshot of the user-interface for word-level annotation.

The structure of the '.txt' file for an example multiscript image is shown in Fig. 1.

The segmented words can further be segmented into its constituent characters or symbols depending on the user's requirement. In most English words, a default left-to-right CC labeling order is sufficient to decompose the word into its constituent characters. However, the lower case letters 'i' and 'j' will require a merging process to group the dot with the base character. For Indic scripts, segmentation of words into individual characters is not trivial. Due to the presence of compound characters (vowel and consonant modifiers), two or more CCs need to be grouped to form a valid character. Some scripts, such as Bangla, Devanagari and Gurmukhi, have a top headline called 'Shirorekha' that interconnects all the characters of a word and will require splitting of symbols. Similarly, cursive text will also require an appropriate splitting process. We employ a polygonal mask to group or split CCs into valid symbols. This allows a straightforward and easy way to segment words into any level of detail such as characters/symbols or unicode symbols depending on the user's requirement. The labeled characters/symbols are assigned a unique label so that they can be accessed by the label without the need to specify the character bounding boxes.

3. WORD-LEVEL ANNOTATION

The schematic flowchart of the word-level annotation toolkit is shown in Figure 2. The action required for each block of the flowchart is also mentioned. The subroutine for tagging a segmented word is separately outlined in the adjacent flowchart. A screenshot of the main menu of the word-level annotation UI is shown in Fig. 3. After the image to be tagged is loaded, one can select a text region using the zoom option at the top of the UI tool bar. Actual segmentation is performed only on the zoomed-in region, accelerating the process. Once a word is zoomed in, the seed points are selected within the character strokes followed by a region-growing process. Whenever the input is through successive mouse click, the last selection should be made with the right click. Canny edges [9] are also shown in a gray shade along with the segmented image for visual comparison of the segmented character boundaries and the edges so as to aid the user in deciding the quality of the segmented output.

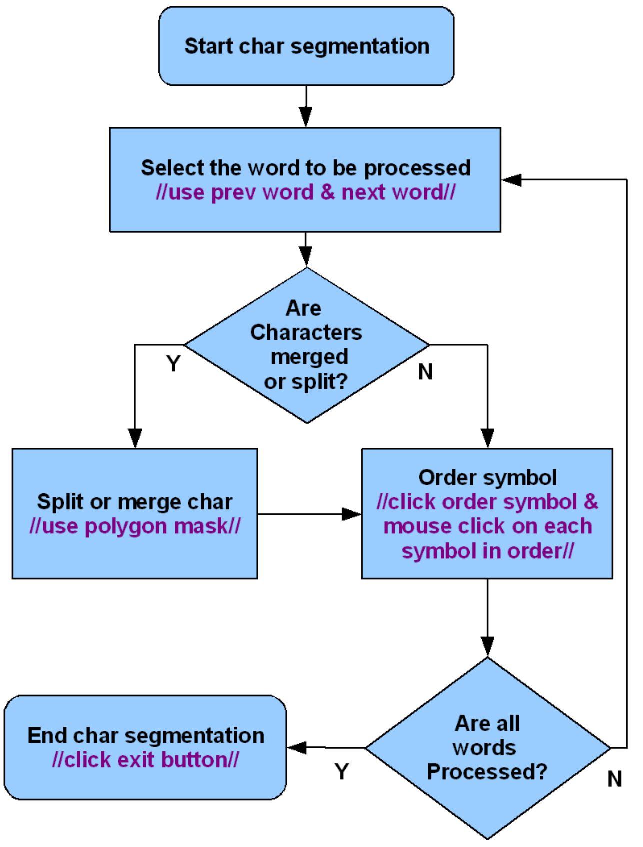
Occasionally, there may be a case where the word is over-segmented or under-segmented depending on the preset threshold value. The color distance threshold value can be varied and perform the region growing process again with the new color distance threshold till we obtain a pixel-accurate segmented image. If the segmentation result is not satisfactory after repeated trials, the user can decide to reload the image and select new seed points and perform a new region growing process. When the text resolution is poor, it may be difficult to place the seed points accurately which can result in a noisy segmentation output. In such a case, a simple thresholding operation may be performed instead of the normal region growing process. The Otsu method [10] is used to binarize the selected gray-scale word block. There is also a provision to invert the binary output for handling text lighter than the background.

Presently, the toolkit has provision for tagging the segmented word image in 10 Indic scripts namely Bangla, Devanagari, Gujarati, Gurmukhi, Kannada, Malayalam, Manipuri, Oriya, Tamil and Telugu besides English. The standard keyboard is used for tagging English text. The word segmentation process and tagging is repeated till all the words are considered.

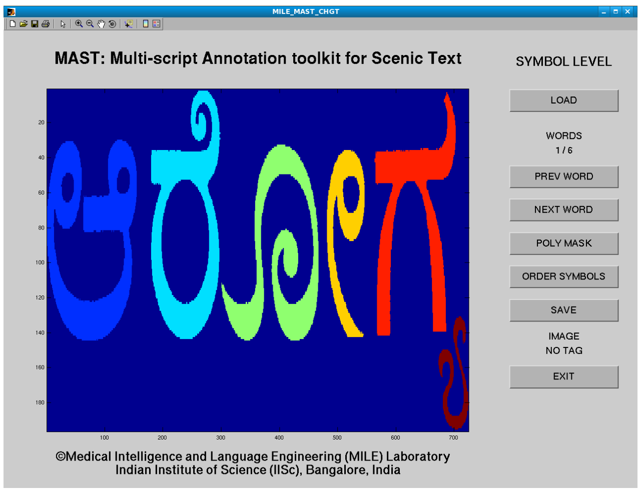
There are other event-driven buttons such as the WORD BOUNDARY button, which when activated, the bounding rectangles of all the previously annotated words are displayed on the image. This facilitates the user to decide whether the word boundaries are accurate or not. It is also used to check if there is any word in the image that is still not annotated. The 'RELOAD' button is used to undo all the changes since last tagging a word.

4. CHARACTER/SYMBOL SEGMENTATION

After all the words have been annotated, each segmented word image may be further segmented into its constituent characters/symbols. When the 'SEGMENT CHAR' button is activated, the user is taken to a new UI where each segmented word is visually examined for possible splitting or merging of symbols to form valid characters. Figure 4 shows the flowchart of the character segmentation process and the main menu of the symbol-level annotation UI. The


Start char segmentation
Select the word to be processed
//use prev word & next word//
Are Characters merged or split?
Y
N
Split or merge char
//use polygon mask//
Order symbol
//click order symbol & mouse click on each symbol in order//
Are all words Processed?
Y
N
End char segmentation
//click exit button//

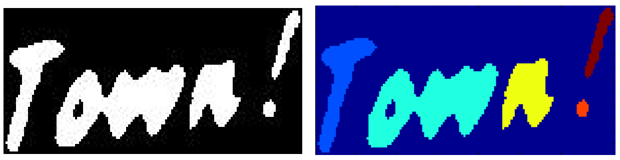
(a)

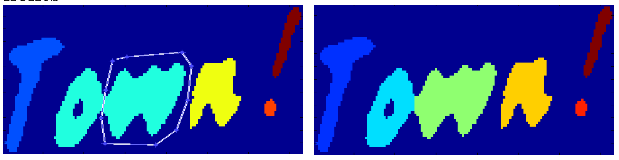

MILE_MAST_CHGT
MAST: Multi-script Annotation toolkit for Scenic Text
SYMBOL LEVEL
LOAD
WORDS
1 / 6
PREV WORD
NEXT WORD
POLY MASK
ORDER SYMBOLS
SAVE
IMAGE
NO TAG
EXIT
©Medical Intelligence and Language Engineering (MILE) Laboratory
Indian Institute of Science (IISc), Bangalore, India

(b)

Figure 4: (a) Flowchart for character/symbol labeling and (b) Screenshot of the user-interface for symbol level annotation.

first segmented word image gets loaded by default. This is followed by labeling process that assigns a unique label to each CC such that the labels are in increasing order from the leftmost to the rightmost CC. The user may split or merge the labeled CCs to form valid characters/symbols using a polygonal mask. Thus, any word can be decomposed into any level of detail such as characters/symbols or uni-


(a) Segmented word image and the initial labeled components


(b) Splitting of merged characters using polygonal mask

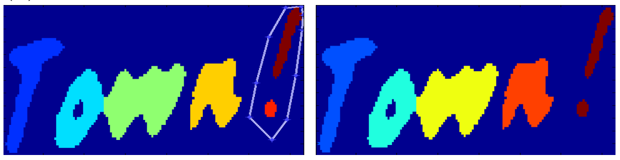

(c) Merging of CCs using polygonal mask to form a compound character

Figure 5: Illustration of labeling individual characters of a word image using polygonal mask.

code symbols depending on the user's requirement. Fig. 5 shows an example of decomposing a segmented image into characters involving a split as well as merge operation using polygonal masks.

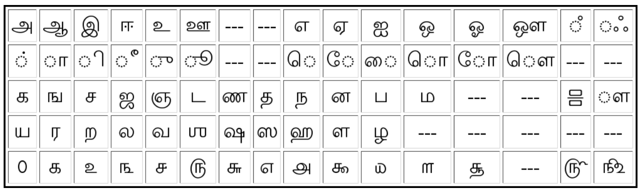
Each of the CCs are re-labeled in left-to-right order after each split/merge operation. While most texts are horizontally aligned, there may be instances where the text is oriented vertically or curved. The automatic labeling of the CCs in a left-to-right fashion fails in such cases. In such a case, the user can manually assign the order of the symbol labels using the mouse. The labeled word image is stored in an uncompressed '.bmp' file format. Unlike other methods [1, 4, 5], our approach allows access to the individual characters by its label without the need to specify the character bounding boxes even when the orientation of the text is inclined, curved or vertical.

5. CREATION OF VIRTUAL KEYBOARD INTERFACE

In addition to segmenting and annotating text present in generic scenic images, MAST can also be used to generate the virtual keyboard interface for any other script. To do this, an image of the keyboard in the desired script is fed to the word-level annotation module as the input.

The keyboard image may be obtained by creating a table in a html file where each cell of the table contains a specific character unicode in any of the scripts available at the Unicode 6.0 Character Code Charts [11]. When this html file is viewed using a web browser, a table with the chosen script fonts will be displayed. A screenshot of this table is used as the virtual keyboard image.

Each key in the keyboard is segmented using the word-level segmentation module and mapped to its corresponding unicode value using the standard English keyboard interface. These keymaps are stored in the resulting '.txt' file. The keyboard image and the corresponding keymaps are placed as



அ	ஆ	இ	ஈ	உ	ஊ	---	---	எ	ஏ	ஐ	ஒ	ஓ	ஔ	◌ஂ	◌ஃ
◌்	◌ா	◌ி	◌ீ	◌ு	◌ூ	---	---	◌ெ	◌ே	◌ை	◌ொ	◌ோ	◌ௌ	---	---
க	ங	ச	ஜ	ஞ	ட	ண	த	ந	ன	ப	ம	---	---	[illegible]	◌ௗ
ய	ர	ற	ல	வ	ஶ	ஷ	ஸ	ஹ	ள	ழ	---	---	---	---	---
௦	௧	௨	௩	௪	௫	௬	௭	௮	௯	௰	௱	௲	---	௹	௺

(a)

```
66
tamil_unicode_1.jpg 8 9 51 45 ENGLISH b85
tamil_unicode_2.jpg 8 59 51 50 ENGLISH b86
tamil_unicode_3.jpg 8 114 51 54 ENGLISH b87
tamil_unicode_4.jpg 8 173 51 50 ENGLISH b88
tamil_unicode_5.jpg 8 228 51 50 ENGLISH b89
tamil_unicode_6.jpg 8 283 51 57 ENGLISH b8a
                         .
                         .
                         .
tamil_unicode_61.jpg 234 511 51 56 ENGLISH bf0
tamil_unicode_62.jpg 234 572 51 63 ENGLISH bf1
tamil_unicode_63.jpg 234 640 51 75 ENGLISH bf2
tamil_unicode_64.jpg 234 720 51 73 ENGLISH bf3
tamil_unicode_65.jpg 234 890 51 48 ENGLISH bf9
tamil_unicode_66.jpg 234 943 51 62 ENGLISH bfa
```

(b)

Figure 6: Illustration of Tamil virtual keyboard interface. (a) The input keyboard image for segmentation and tagging each key with the corresponding unicode (b) The .txt file generated using MAST which contain the keymaps for each key.

required in the toolkit which can be used for ground truthing text in the chosen script.

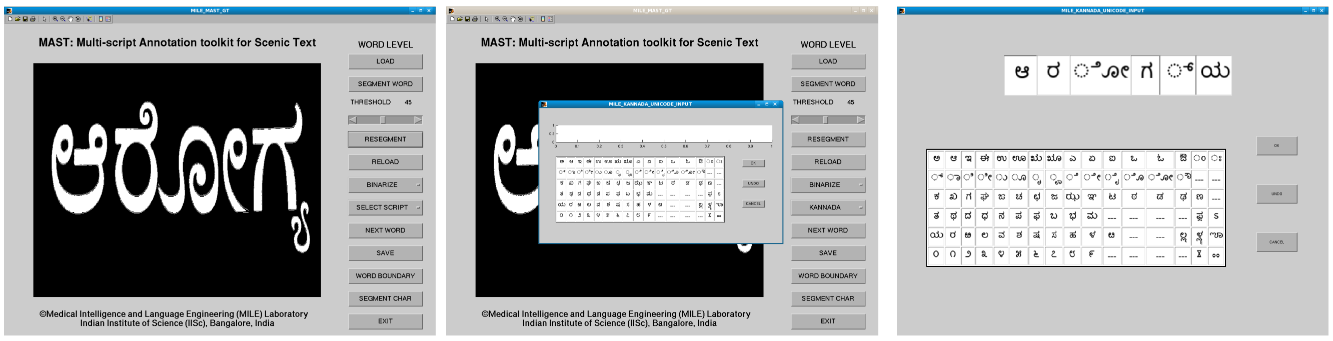
For example, in Fig. 6(a) all the 66 keys of the Tamil keyboard are segmented individually and mapped to its corresponding unicode. The tagged file containing the Tamil unicode values is shown 6(b). The virtual keyboard image and the associated '.txt' file are then used to create the interface for ground truthing texts in Tamil.

6. RESULTS

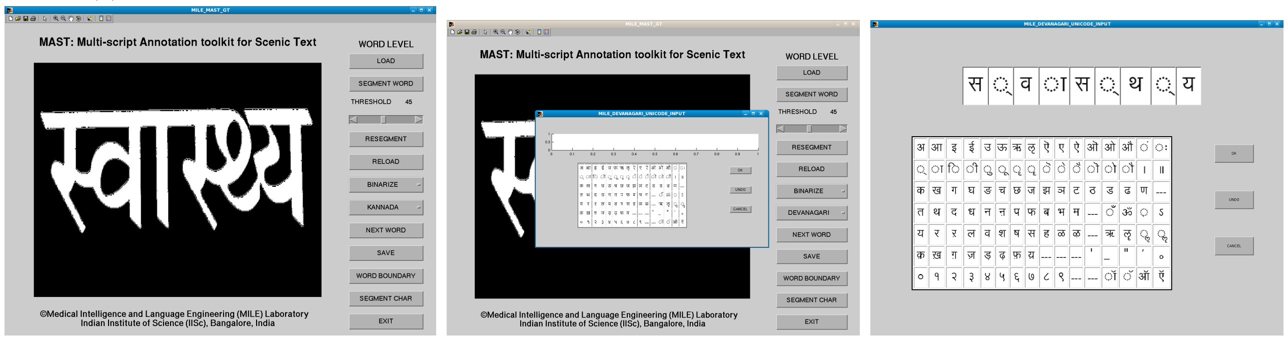
Fig. 7 illustrates the word-level annotation process the multi-script scene image shown in Figure 1(a). Fig. 7(a) shows a segmented Kannada word image, the virtual keyboard interface for entering Kannada text and the annotation as entered by the tagger. Fig. 7(b) and (c) show the results of segmenting and tagging Devanagiri and English words respectively. The standard keyboard is used for entering the text.

Fig. 8 shows some examples of pixel-level ground truth obtained using MAST for typical scene images containing multiple scripts and various types of text layouts. It may be noted that bounding box-based ground truth are not suitable for such images that contain arbitrarily-oriented text.

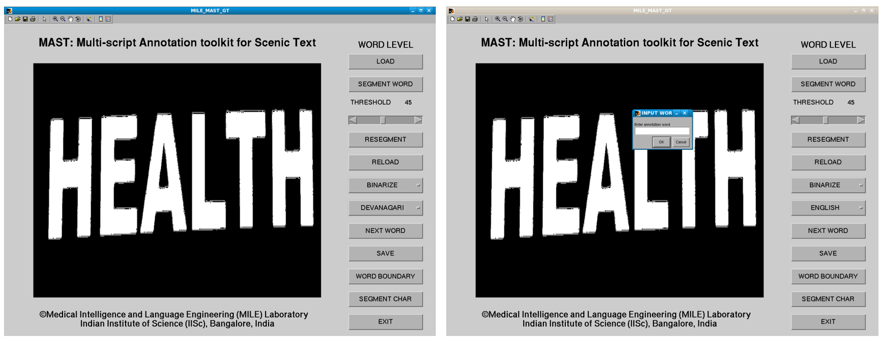
The flexibility of the toolkit is illustrated in Fig. 9 which shows words in different scripts having being segmented into their constituent characters/symbols. Broken characters are merged into valid characters while touching characters, cursive text or compound characters are appropriately split and labeled. Clearly, the method can cater to any orientation of text, writing style and script.


MAST: Multi-script Annotation toolkit for Scenic Text
WORD LEVEL
LOAD
SEGMENT WORD
THRESHOLD 45
RESEGMENT
RELOAD
BINARIZE
SELECT SCRIPT
KANNADA
NEXT WORD
SAVE
WORD BOUNDARY
SEGMENT CHAR
EXIT
©Medical Intelligence and Language Engineering (MILE) Laboratory
Indian Institute of Science (IISc), Bangalore, India
ಆರೋಗ್ಯ

(a) A segmented Kannada word image and annotated unicode text using the virtual keyboard interface


MAST: Multi-script Annotation toolkit for Scenic Text
WORD LEVEL
LOAD
SEGMENT WORD
THRESHOLD 45
RESEGMENT
RELOAD
BINARIZE
KANNADA
DEVANAGARI
NEXT WORD
SAVE
WORD BOUNDARY
SEGMENT CHAR
EXIT
©Medical Intelligence and Language Engineering (MILE) Laboratory
Indian Institute of Science (IISc), Bangalore, India
स्वास्थ्य

(b) Annotation of segmented Devanagari word image


MAST: Multi-script Annotation toolkit for Scenic Text
WORD LEVEL
LOAD
SEGMENT WORD
THRESHOLD 45
RESEGMENT
RELOAD
BINARIZE
DEVANAGARI
ENGLISH
NEXT WORD
SAVE
WORD BOUNDARY
SEGMENT CHAR
EXIT
©Medical Intelligence and Language Engineering (MILE) Laboratory
Indian Institute of Science (IISc), Bangalore, India
HEALTH

(c) Annotation of a segmented English word image using the standard keyboard interface

Figure 7: Illustration of word-level annotation for the multi-script scene image shown in Fig 1(a).

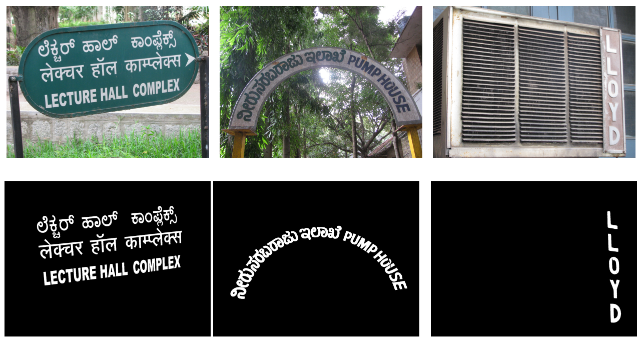
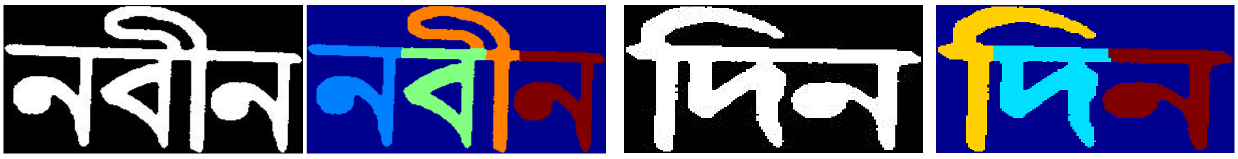


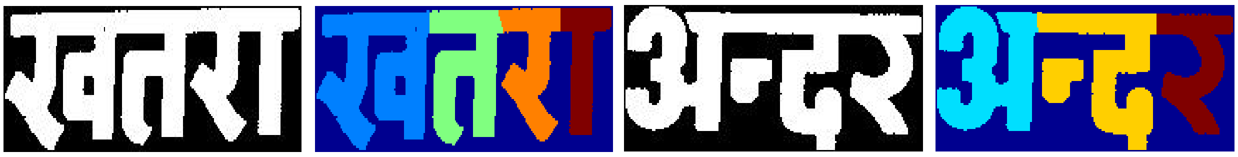
Figure 8: Sample ground truth segmented images obtained using MAST for scene images with multi-script content and arbitrary text orientations.

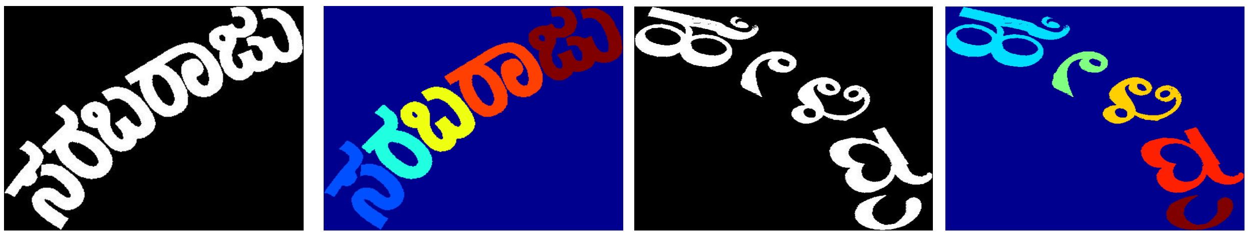
7. CONCLUSION AND FUTURE WORK

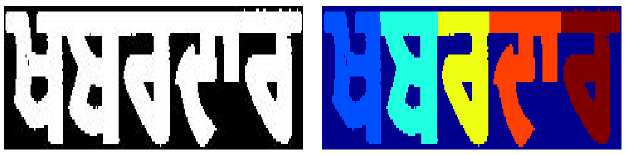
We have presented a versatile software tool for annotating text present in multi-script scenic images. It can be used to create ground truth data for any generic image at the word level or character/symbol level depending on the user's requirement. The ground truth text regions are represented at the pixel-level. Bounding boxes of adjacent CCs can have a high degree of overlap for skewed or curved text and hence, it does not represent an accurate location of the characters. The pixel-level ground truth gives an accurate representation of the text regions for arbitrary text orientations. Moreover, The polygonal mask, used in our approach, for symbol-level segmentation allows a straightforward and easy way to extract characters/symbols for training various recognition algorithms. The pixels are indexed by a unique label that allows direct extraction of a character/symbol without specifying its bounding box.

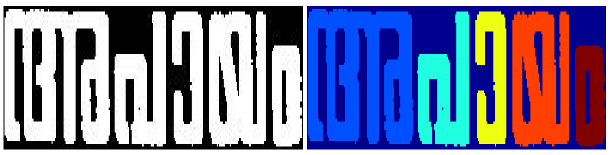
Currently, we have provision for tagging 10 Indic scripts and English. The Devanagari virtual keyboard interface can

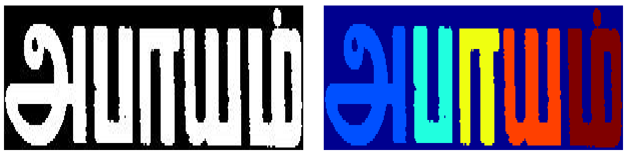

(a) Bangla

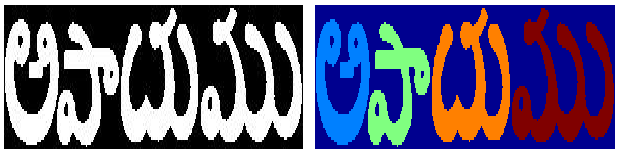

(b) Devanagari

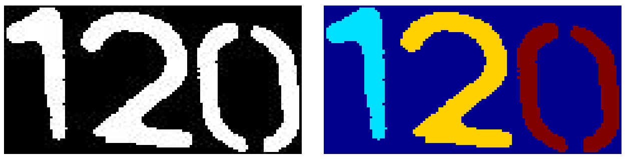

(b) Kannada

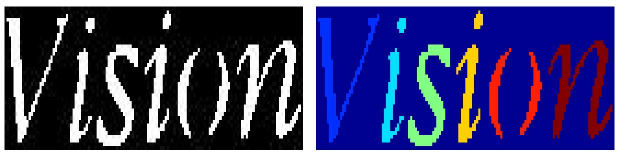

(c) Gurmukhi


(d) Malayalam


(e) Tamil


(f) Telugu


(g) Indo-Arabic numerals


(h) English

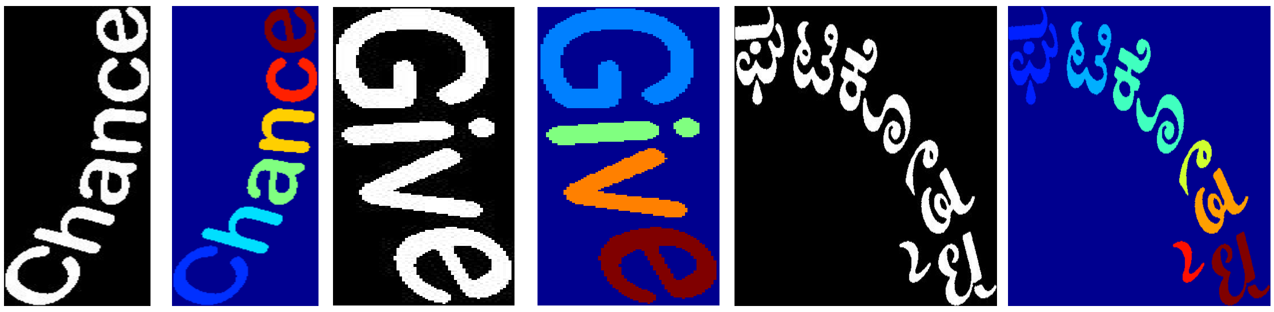

(b) Example word images for which manual ordering of the character labels is required

Figure 9: Sample outputs of symbol-level segmentation for various scripts. Note that merged/broken characters are split/grouped into valid symbols.

be used to tag other scripts like Marathi, Hindi, Konkani and Sanskrit since all of them have similar scripts. Likewise, the Bangla virtual keyboard interface can also be used to tag Assamese text. One useful contribution of the paper is the design of virtual keyboard interface. It is easy to create interfaces for other scripts thereby making the toolkit applicable to any script. The software is available online at [12] along with a detailed description of the functionalities of each of the menu items. We hope that researchers worldwide will find it useful in creating ground truth database for any generic document image.

Owing to its simplicity, we have chosen Otsu binarization for segmenting small text. In future, the binarization module can be replaced with a more sophisticated method such as color clustering. Support for tagging other entities such as logos, symbols and graphic objects can also be added. Using the toolkit, we are building a database containing multi-script scene images which will be made publicly available. We plan to groundtruth other existing databases at the pixel-level. We also plan to provide web interface for users to upload and tag their own images, which can help create a large database.

8. REFERENCES

[1] ICDAR 2003 Robust reading competition data set, http://algoval.essex.ac.uk/icdar/Competitions.html.

[2] C. Wolf and J.M Jolion, Object count/area graphs for the evaluation of object detection and segmentation algorithms, Intl. Jl. Document Analysis and Recognition, 8(4), 280 - 296, 2006.

[3] ICDAR 2011 Robust Reading Competiton, Challenge 1: "Reading Text in Born-Digital Images (Web and Email)", http://www.cvc.uab.es/icdar2011competition/

[4] The Chars74K dataset, available at http://www.ee.surrey.ac.uk/CVSSP/demos/chars74k/

[5] KAIST Scene Text Database, http://www.iapr-tc11.org/mediawiki/index.php/KAIST_Scene_Text_Database

[6] F. Shafait, D. Keysers, and T. Breuel, Pixel-accurate representation and evaluation of page segmentation in document images, Proc. Intl. Conf. Pattern Recognition, 872–875, 2006.

[7] H. Baird, M. Mill, and C. An, Truthing for pixel-accurate segmentation, Proc. Intl. Workshop Document Analysis and Systems, 379-385, 2008.

[8] E. Saund, J. Lin and P Sarkar, PixLabeler: User Interface for Pixel-Level Labeling of Elements in Document Images, Proc. Intl. Conf. Document Analysis and Recognition, 646,650, 2009.

[9] J Canny: A computational approach to edge detection, IEEE Trans. Pattern Analysis and Machine Intelligence, 8(6), 679 - 698, 1986.

[10] N Otsu, A threshold selection method from gray-level histograms, IEEE Trans. Systems Man Cybernetics, 9(1), 62 - 66, 1979.

[11] Unicode 6.0 Character Code Charts, http://unicode.org/charts/

[12] MAST: Multi-script annotation toolkit for scenic text, http://mile.ee.iisc.ernet.in/mast